

คู่มือติดตั้ง Local AI n8n และ ComfyUI บน Mac mini

คู่มือภาษาไทยสำหรับ Mac mini ชิป Apple Silicon ตั้งแต่ M1 ขึ้นไป เหมาะกับผู้เริ่มต้นที่ต้องการให้ AI เขียนข้อความ ทำงานอัตโนมัติ และสร้างภาพบนเครื่องเดียวกัน

เครื่องมือ	หน้าที่	ติดตั้งที่ไหน
Ollama	รันโมเดลภาษา เช่น ช่วยเขียนและสรุป	แอปบน macOS
n8n	เชื่อมต่อขั้นตอนและเรียกใช้ AI อัตโนมัติ	Docker Desktop
ComfyUI	สร้างภาพจากพรมบัพด้วยโมเดลในเครื่อง	Comfy Desktop บน macOS

ผลลัพธ์เมื่อทำครบ

คุณจะคุยกับ AI ในเครื่อง เปิด n8n ผ่านเบราว์เซอร์ สร้างภาพแรก และทดลอง workflow ที่ให้ AI เขียนพรมบัพก่อนส่งไปสร้างภาพได้

เส้นทางในคู่มือ

ตรวจเครื่อง → ติดตั้ง Ollama → ทดสอบ AI → ติดตั้ง Docker และ n8n → เชื่อม n8n กับ AI → ติดตั้ง ComfyUI → สร้างภาพ → เชื่อมครบสามตัว

เอกสารตรวจสอบวันที่ 3 ตุลาคม 2026 ชื่อเมนูอาจต่างกันตามรุ่นของแอป คู่มือนี้ใช้บริการ Local และ workflow ที่ก่อดันเองเป็นหลัก

ใช้เน็ตสำหรับดาวน์โหลดแอปและโมเดลครั้งแรก เมื่อเตรียมครบ ตัวอย่าง AI และภาพสามารถประมวลผลในเครื่องได้ บริการภายนอกที่นำมาเชื่อมเพิ่มยังต้องใช้อินเทอร์เน็ต

คำแนะนำเรื่อง RAM พื้นที่ว่าง และค่าทดสอบในคู่มือเป็นจุดเริ่มต้นเชิงปฏิบัติ ไม่ใช่ผล benchmark หรือการรับประกันความเร็ว

อ้างอิง [1], [4], [6], [13]

1 ตรวจ Mac mini และเตรียมเครื่อง

เปิดเมนู Apple → About This Mac แล้วดู Chip และ Memory ถ้าเป็น M1 หรือใหม่กว่า ให้ใช้เส้นทางนี้ได้
ถ้าเป็น Intel คู่มือ Comfy Desktop ส่วนนี้ใช้ไม่ได้ ส่วน Ollama อาจรันด้วย CPU ได้

อัปเดต macOS ให้เป็นรุ่นที่แอปทั้งหมดรองรับ Ollama ต้องการ macOS 14 ขึ้นไป ส่วน Docker Desktop
รองรับ macOS รุ่นปัจจุบันและสองรุ่นหลักก่อนหน้า ตรวจหน้าดาวน์โหลดก่อนติดตั้ง

RAM ในเครื่อง	แนวทางเริ่มต้น
8 GB	ลอง AI ขนาดเล็กและ n8n ก่อน งานภาพอาจติดข้อจำกัดหน่วยความจำ
16 GB	เริ่ม Qwen3 4B และ SD 1.5 ขนาด 512 × 512 ที่ละงาน
24 ถึง 32 GB	มีพื้นที่ให้เปิดแอปอื่นเพิ่ม แต่ยังคงเริ่มจากโมเดลเล็ก
48 GB ขึ้นไป	ค่อยทดลองโมเดลหรือภาพที่ใหญ่ขึ้นหลังชุดเริ่มต้นผ่าน

แนะนำพื้นที่ว่าง 30 ถึง 50 GB สำหรับชุดเริ่มต้นและไฟล์ชั่วคราว ถ้าจะเก็บหลายโมเดลควรเผื่อมากกว่านี้
ขนาดไฟล์โมเดลไม่เท่ากับ RAM ที่ใช้จริง

เปิด Terminal ด้วย Spotlight โดยกด Command + Space แล้วพิมพ์ Terminal คัดลอกเฉพาะคำสั่งใน
กรอบ ไม่ต้องพิมพ์เครื่องหมาย \$ เพิ่ม

```
uname -m  
sw_vers -productVersion  
df -h ~
```

ผล uname -m ควรเป็น arm64 ถ้าเป็น x86_64 ทั้งที่เครื่องเป็นชิป M ให้ปิด Terminal แล้วเปิดแบบปกติ
โดยไม่ใช้ Open using Rosetta

เส้นทางหลักใช้แอปติดตั้งสำเร็จรูป จึงไม่ต้องติดตั้ง Homebrew, Node.js หรือ Python แยกเพื่อทำ
ตัวอย่างในคู่มือ

อ้างอิง [1], [5], [13]

2 ติดตั้ง Ollama และโหลด Local AI

เข้าเว็บไซต์ Ollama เลือกดาวน์โหลดสำหรับ macOS เปิดไฟล์ .dmg แล้วลาก Ollama ไปที่ Applications

ดาวน์โหลด Ollama สำหรับ Mac

เปิด Ollama จาก Applications ถ้ามีหน้าต่างขอเพิ่มคำสั่ง ollama ใน Terminal ให้ดำเนินการตามหน้าต่างนั้น แล้วเปิด Terminal ใหม่

```
ollama --version
```

ควรเห็นหมายเลขเวอร์ชัน ถ้ายังไม่พบคำสั่ง ให้ดูหัวข้อแก้ปัญหา อย่าเปิด ollama serve ซ้ำถ้าแอป Ollama กำลังเปิดอยู่แล้ว

ดาวน์โหลดโมเดลตัวอย่างที่เลือกไว้ ชื่อรุ่นนี้เป็นตัวอย่างที่ขนาดไม่ใหญ่ ไม่ได้อ้างว่าเป็นรุ่นล่าสุดหรือดีที่สุด

```
ollama pull qwen3:4b
ollama list
ollama run qwen3:4b
```

หน้าโมเดลระบุไฟล์ Qwen3 4B แบบ Q4_K_M ประมาณ 2.5 GB ณ วันที่ตรวจสอบ ต้องเผื่อ RAM สำหรับบริบทและ macOS เพิ่มจากขนาดไฟล์ด้วย

เมื่อมีช่องให้พิมพ์ ลองถามว่า ช่วยอธิบาย Local AI เป็นภาษาไทยสำหรับมือใหม่ 3 บรรทัด แล้วกด Enter เมื่อคุยเสร็จพิมพ์ /bye

```
ollama ps
```

ตรวจสอบ PROCESSOR ของโมเดลที่กำลังโหลดอยู่เพื่อการใช้ GPU หรือ CPU ถ้าไม่มีรายการให้ถาม AI อีกครั้งแล้วตรวจทันที

ผ่านขั้นนี้เมื่อ ollama list แสดง qwen3:4b และ AI ตอบข้อความได้ Ollama ใช้พอร์ต 11434 โดยปกติ และฟังเฉพาะเครื่องตัวเองที่ 127.0.0.1

อ้างอิง [1], [2], [4]

3 ทดสอบ Local AI ผ่าน API

API คือช่องทางให้โปรแกรมอื่นเรียก AI ได้ ขั้นนี้จะพิสูจน์ว่า n8n สามารถใช้บริการเดียวกับที่เราคุยใน Terminal ได้

ตรวจรายชื่อโมเดลผ่าน API

```
curl -fsS http://127.0.0.1:11434/api/tags
```

ส่งคำถามแบบได้คำตอบก่อนเดียว คัดลอกคำสั่งทั้งกรอบลง Terminal

```
curl -fsS http://127.0.0.1:11434/api/chat \  
-H 'Content-Type: application/json' \  
-d '{  
  "model": "qwen3:4b",  
  "messages": [{  
    "role": "user",  
    "content": "อธิบาย Local AI เป็นภาษาไทย 3 บรรทัด"  
  }],  
  "stream": false,  
  "think": false,  
  "keep_alive": 0,  
  "options": {"num_ctx": 4096, "num_predict": 256}  
}'
```

ผลลัพธ์ควรเป็น JSON ที่มี message.content เป็นคำตอบ และ done เป็น true ถ้าได้ model not found ให้ตรวจชื่อด้วย ollama list

ค่าที่ใช้	ความหมาย
stream false	รอคำตอบครบก่อนส่ง JSON กลับ ทำให้เริ่มใช้ใน n8n ง่าย
think false	ปิดโหมดคิดของ Qwen3 ในตัวอย่าง เพื่อตอบสนองและควบคุมงาน
keep_alive 0	ปล่อยโมเดลออกจากหน่วยความจำหลังตอบ เพื่อเตรียมงานภาพ
num_ctx 4096	เริ่มจากบริบทไม่ยาว เพื่อลดภาระหน่วยความจำ

ถ้าจะใช้เฉพาะ Local ให้ตรวจว่าเลือกโมเดลที่ดาวน์โหลดไว้ ไม่เลือก cloud model และไม่เปิดเครื่องมือค้นเว็บโดยไม่ตั้งใจ สามารถปิด Cloud features ตาม Ollama FAQ ได้

อ้างอิง [3], [4]

4 ติดตั้ง Docker Desktop สำหรับ n8n

n8n อยู่ใน Docker ส่วน Ollama และ ComfyUI อยู่บน macOS เพื่อใช้ GPU ของ Apple โดยตรง Ollama ใน Docker Desktop บน Mac ไม่ได้ใช้ GPU passthrough แบบเครื่อง NVIDIA

ดาวน์โหลด Docker Desktop สำหรับ Mac with Apple silicon เปิด Docker.dmg แล้วลากแอปไปที่ Applications เปิดแอปและทำขั้นตอนเริ่มต้นจน Docker Engine พร้อม

ดาวน์โหลดและดูเงื่อนไข Docker Desktop

ตรวจสอบเงื่อนไขการใช้ Docker Desktop ตามประเภทองค์กรของคุณ การใช้งานส่วนตัว การศึกษา และธุรกิจ เล็กที่เข้าเกณฑ์ใช้ได้ฟรี องค์กรบางประเภทต้องมี subscription

เส้นทางนี้ต้องใช้ Docker Desktop 4.34 ขึ้นไป ลงชื่อเข้าใช้บัญชี Docker ตามหน้าตั้งค่า แล้วเปิด Settings → Resources → Network → Enable host networking → Apply and restart

Host networking ทำให้ n8n ใน container เรียกบริการบน Mac ผ่าน 127.0.0.1 ได้ ตัวเลือกนี้ใช้ร่วมกับ Enhanced Container Isolation ไม่ได้ ถ้าเครื่องถูกองค์กรบังคับเปิด ECI ให้ผู้ดูแลเลือกเครือข่ายที่เหมาะสมกับนโยบายก่อนใช้ไฟล์นี้

เปิด Terminal ใหม่และตรวจสอบว่า Docker Engine กับ Compose พร้อม

```
docker version
docker compose version
```

docker version ควรมีทั้ง Client และ Server ถ้ามีเฉพาะ Client หรือขึ้น Cannot connect ให้เปิด Docker Desktop และรอ Engine พร้อม

สำหรับ RAM 16 GB อาจเริ่มจัด RAM ให้ Docker ราว 2 ถึง 4 GB แล้วดู Activity Monitor เพื่อปรับตามงาน ตัวอย่างนี้ใช้ n8n ตัวเดียว ไม่มีชุด Assistant และ sandbox เพิ่มเติม

อ้างอิง [4], [5], [6]

5 สร้าง n8n ด้วย Docker Compose

สร้างโฟลเดอร์สำหรับการตั้งค่า แล้วเปิดไฟล์ `compose.yaml` ด้วย `nano`

```
mkdir -p ~/local-ai-stack
cd ~/local-ai-stack
nano compose.yaml
```

วางข้อความด้านล่างลงใน `nano` บันทึกด้วย `Control + O` แล้ว `Enter` จากนั้นออกด้วย `Control + X` หากมีไฟล์ตัวอย่างใน ZIP ใช้ไฟล์ `compose.yaml` นั้นได้เลย

```
name: mac-local-ai
services:
  n8n:
    image: n8nio/n8n:2.41.6
    restart: unless-stopped
    network_mode: host
    environment:
      N8N_LISTEN_ADDRESS: 127.0.0.1
      N8N_HOST: localhost
      N8N_PORT: 5678
      N8N_PROTOCOL: http
      N8N_EDITOR_BASE_URL: http://localhost:5678
      GENERIC_TIMEZONE: Asia/Bangkok
      TZ: Asia/Bangkok
      N8N_ENFORCE_SETTINGS_FILE_PERMISSIONS: "true"
      N8N_DIAGNOSTICS_ENABLED: "false"
      N8N_PERSONALIZATION_ENABLED: "false"
    volumes:
      - n8n_data:/home/node/.n8n
volumes:
  n8n_data:
    name: mac-local-ai-n8n-data
```

เลือก `n8n 2.41.6` ตามรุ่น `stable` ที่หน้าเอกสารฉบับวันที่ตรวจสอบ เพื่อให้ตัวอย่างระบุเวอร์ชันชัดเจน ใช้ `SQLite` และเก็บข้อมูลใน volume ชื่อ `mac-local-ai-n8n-data`

`network_mode host` ต้องเปิด `Host networking` ตามบทความ ไม่ใส่ `ports` เพิ่มในโหนดนี้ ตัวอย่างกำหนด `n8n` ให้ฟัง `127.0.0.1` เพื่อใช้งานภายในเครื่อง

ไฟล์ตัวอย่าง `compose.yaml`

อ้างอิง [6], [7], [8], [9]

6 เปิด n8n และเชื่อมกับ Ollama

จากโฟลเดอร์ ~/local-ai-stack รันคำสั่งต่อไปนี้ รอให้โหลด image และเริ่มบริการ

```
docker compose config
docker compose up -d
docker compose ps
docker compose logs --tail=50 n8n
```

เปิด <http://localhost:5678> ในเบราว์เซอร์ สร้างบัญชีเจ้าของสำหรับ n8n ในเครื่อง แล้วเก็บรหัสผ่านไว้

เปิด n8n ในเครื่อง

พิสูจน์การเชื่อมต่อจากใน container โดยเปิดแอป Ollama ไว้ก่อน แล้วรันคำสั่งนี้

```
docker compose exec n8n node -e '
fetch("http://127.0.0.1:11434/api/tags")
  .then(r => {
    if (!r.ok) throw Error(r.status);
    return r.json();
  })
  .then(console.log)
  .catch(e => {
    console.error(e.message);
    process.exit(1);
  })
'
```

ควรเห็นข้อมูลโมเดล qwen3:4b ถ้า curl จาก Mac ผ่าน แต่คำสั่งนี้ไม่ผ่าน ให้ตรวจ Host networking และรีสตาร์ท Docker ก่อนแก้ค่า Ollama

ดาวน์โหลด n8n-01-local-caption.json แล้วเลือก Import from File ในเมนู workflow ของ n8n กด Execute workflow เปิดผลของ Ask Ollama แล้วดู message.content

ไฟล์ตัวอย่าง n8n-01-local-caption.json

workflow นี้ใช้ Manual Trigger และ HTTP Request เท่านั้น กดรันเอง ไม่ส่งข้อมูลไปบริการภายนอก และไม่ต้องมี API key ของผู้ให้บริการ AI

ถ้าต้องการใช้ Ollama Chat Model node ภายหลัง ให้ตั้ง Base URL เป็น <http://127.0.0.1:11434> โดย Local Ollama ปกติไม่ต้องกรอก API key แล้วเลือกโมเดลที่โหลดไว้

อ้างอิง [6], [10], [11], [12]

7 สร้าง workflow Local AI ด้วยตัวเอง

หากอยากเข้าใจการตั้งค่า ให้สร้าง workflow ใหม่ เพิ่ม Manual Trigger แล้วต่อกับ HTTP Request ตั้งค่าตามตารางนี้

ช่องใน HTTP Request	ค่าที่กรอก
Method	POST
URL	http://127.0.0.1:11434/api/chat
Authentication	None
Send Body	เปิด
Body Content Type	JSON
Specify Body	Using JSON
Timeout ใน Options	300000 มิลลิวินาที

```
{
  "model": "qwen3:4b",
  "messages": [{
    "role": "user",
    "content": "เขียนแคปชั่นไทย 3 บรรทัด ขวนมือใหม่ลอง Local AI"
  }],
  "stream": false,
  "think": false,
  "keep_alive": 0,
  "options": {"num_ctx": 4096, "num_predict": 256}
}
```

กด Execute workflow แล้วเปิด Output ของ HTTP Request คำตอบอยู่ที่ message.content เปลี่ยนข้อความ content ได้ตามงาน เช่น สรุปบันทึกการประชุม เขียนอีเมล หรือจัดหมวดหมู่ข้อความ

ถ้าจะอ่านคำตอบใน node ถัดไป ใช้ expression {{ \$json.message.content }} อย่าอ่าน response ซึ่งเป็นชื่อฟิลด์ของ API คนละแบบ

ทดสอบด้วยข้อความสั้นก่อน แล้วค่อยเพิ่มข้อมูลจริง ตั้งค่า timeout ให้พอสำหรับโหนดโมเดลรอบแรก

อ้างอิง [3], [10]

8 ติดตั้ง ComfyUI ผ่าน Comfy Desktop

เข้าเว็บไซต์ Comfy ดาวน์โหลดแอปสำหรับ macOS Apple Silicon เปิด .dmg ลาก Comfy Desktop ไป Applications แล้วเปิดแอป

ดาวน์โหลด Comfy Desktop

ในหน้า Chooser เลือก + New Instance หรือ New Install ตามรุ่นของแอป เลือก Standalone หรือ Local ตั้งชื่อ Local Images และเลือกโฟลเดอร์บน SSD ภายในเครื่อง

Comfy Desktop จะจัดการ Python และ dependencies ให้ รอนจนติดตั้งเสร็จแล้วเปิด instance ที่สร้าง อย่าเลือก Comfy Cloud หรือ Remote หากต้องการประมวลผลบน Mac เครื่องนี้

ตรวจโฟลเดอร์โมเดลจริงผ่าน Desktop Settings → Storage หรือเมนูแคปซูลด้านบน → Storage กด ไอคอนโฟลเดอร์ของ Shared Models ที่เป็น Primary

ใช้ตำแหน่งที่แอปแสดงเป็นหลัก ในโฟลเดอร์ models จะมีโฟลเดอร์ย่อย checkpoints สำหรับโมเดลภาพ แบบ checkpoint จด Output Directory ไว้ด้วยเพื่อหาภาพหลังสร้างเสร็จ

ดาวน์โหลด SD 1.5 FP16 ตามลิงก์จากเอกสาร Comfy แล้ววางไฟล์ลง checkpoints

ดาวน์โหลด v1-5-pruned-emaonly-fp16.safetensors

ไฟล์ต้องลงท้าย .safetensors ไม่ใช่หน้าเว็บ .html หรือไฟล์ทางลัด ถ้าโหลดผ่านหน้า Hugging Face ให้ กดปุ่มดาวน์โหลดไฟล์จริง

```
models/checkpoints/  
v1-5-pruned-emaonly-fp16.safetensors
```

ตัวอย่างนี้ใช้ SD 1.5 เพื่อเริ่มจากงานภาพขนาดเล็ก เมื่อสำเร็จแล้วค่อยทดลองโมเดลที่ใหญ่ขึ้น อ่าน model card และเงื่อนไขใช้โมเดลก่อนนำไปใช้

อ้างอิง [13], [14], [15], [17], [20]

9 สร้างภาพแรกและตรวจ ComfyUI API

โหลด default text-to-image workflow หรือ workflow SD 1.5 ที่ใช้ node มาตรฐาน ถ้าไม่ได้แสดงโดยอัตโนมัติ ใช้ตัวอย่าง text-to-image.json จากหน้า First Generation ของ Comfy

เปิดคู่มือภาพแรกและไฟล์ workflow ทางการ

ใน Load Checkpoint เลือก v1-5-pruned-emaonly-fp16.safetensors ถ้าไม่พบ ให้ Refresh หรือปิดแล้วเปิด instance ใหม่

ตำแหน่ง	ค่าเริ่มต้นสำหรับทดสอบ
Positive prompt	a cup of matcha latte, wooden table, soft natural light, product photography
Negative prompt	blurry, low quality, watermark, text
Empty Latent Image	width 512, height 512, batch_size 1
KSampler	steps 20, cfg 7, sampler euler, scheduler normal, denoise 1

กด Run รอภาพปรากฏใน Save Image แล้วดูไฟล์ที่ Output Directory อย่าตัดสินว่าเครื่องค้างเพียงเพราะรอบแรกโหลดโมเดลนาน

ก่อนเชื่อม n8n ให้ดู host และ port จริงใน Manage ของ instance หรือ log เริ่มต้น สำหรับตัวอย่างไฟล์นี้ให้ตั้ง Launch arguments เป็นค่าด้านล่าง แล้วรีสตาร์ท instance

```
--listen 127.0.0.1 --port 8188
```

หากมี --listen หรือ --port เดิม ให้แก้ค่าเดิม ไม่เพิ่มซ้ำ Desktop บางรุ่นอาจเลือกพอร์ตอื่น ให้ใช้ค่าที่เปิดได้จริง และปรับ URL ทุก node ใน workflow ให้ตรงกัน

```
curl -fsS http://127.0.0.1:8188/system_stats
```

ควรได้ JSON ของระบบและอุปกรณ์ ตรวจสอบว่ามีอุปกรณ์ mps สำหรับการใส่ GPU ของ Apple ถ้า API ไม่ตอบ ให้ตรวจพอร์ตและว่า instance เปิดอยู่

อ้างอิง [14], [15], [16]

10 ให้ n8n ส่ง Local AI และสร้างภาพ

ตัวอย่างที่สองเชื่อมทั้งสามเครื่องมือ n8n ส่งคำสั่งให้ Ollama เขียนพรอมป์ภาษาอังกฤษ แล้วส่งพรอมป์ไปยัง ComfyUI เมื่อภาพเสร็จ n8n ดึงไฟล์ภาพกลับมา

เปิด Ollama, Docker Desktop และ ComfyUI instance ตรวจสอบว่าโหลด qwen3:4b และ SD 1.5 ตามชื่อที่ระบุไว้แล้ว

ดาวน์โหลด n8n-02-local-ai-to-image.json แล้ว Import from File ใน n8n ถ้า ComfyUI ใช้พอร์ตอื่น ให้แก้ URL ใน Queue Image, Read History และ Download Image ให้ตรงกัน

ไฟล์ตัวอย่าง n8n-02-local-ai-to-image.json

Node	สิ่งที่ทำ
Write Image Prompt	Ollama เขียนพรอมป์และปล่อยโมเดลออกจาก RAM เมื่อเสร็จ
Queue Image	POST /prompt ส่งกราฟ SD 1.5 และรับ prompt_id
Wait และ Read History	รอ 5 วินาที แล้ว GET /history/{prompt_id}
Image Ready และ Retry Allowed	วนตรวจสอบสูงสุด 60 ครั้ง หรือหยุดเมื่อระบบรายงาน error
Download Image	GET /view แล้วเก็บภาพเป็น binary field ชื่อ data

กด Execute workflow ผลสำเร็จคือ Download Image มีแท็บ Binary ที่เปิดหรือดาวน์โหลดภาพได้ และ ComfyUI มีภาพใน Output Directory

กราฟตัวอย่างใช้ SaveImage node ID 9 และไฟล์ checkpoint ตามชื่อในคู่มือ ถ้าสร้างกราฟใหม่ ต้องแก้ชื่อ checkpoint และ node ID ที่อ่านผลให้ตรงกับกราฟนั้น

ระบบส่งงานเป็นคิว ไม่ได้คืนภาพทันทีเมื่อ POST /prompt ตัวอย่างตรวจราว 5 นาที หาก timeout งานใน ComfyUI อาจยังทำอยู่ให้ดูคิวก่อนกดรันซ้ำและหยุดงานเก่าจาก ComfyUI หากจำเป็น

เปลี่ยนไอเดียภาพได้ที่ content ใน Write Image Prompt ตัวอย่างใช้ seed คงที่ 12345 และ batch 1 เพื่อให้เริ่มจากงานเล็ก

อ้างอิง [3], [10], [16]

11 เรียก ComfyUI API โดยตรง

ไฟล์ comfy-sd15-api.json ในชุดตัวอย่างมีกราฟครบสำหรับสร้างภาพ SD 1.5 ใช้ทดสอบการส่งงานโดยไม่ผ่าน Ollama ได้

ไฟล์ตัวอย่าง comfy-sd15-api.json

นำไฟล์มาไว้ใน ~/local-ai-stack จากนั้นส่งงานจาก Terminal

```
cd ~/local-ai-stack
curl -fsS http://127.0.0.1:8188/prompt \
  -H 'Content-Type: application/json' \
  --data-binary @comfy-sd15-api.json
```

ควรได้ prompt_id และ number ถ้าได้ error หรือ node_errors ให้ตรวจสอบชื่อ checkpoint และชนิด node ก่อนส่งซ้ำ

คัดลอก prompt_id จากผลก่อนหน้าแทน YOUR_PROMPT_ID แล้วตรวจสอบเมื่อภาพเสร็จ

```
curl -fsS 'http://127.0.0.1:8188/history/YOUR_PROMPT_ID'
```

ระหว่างรออาจได้ {} เมื่อเสร็จให้ดู outputs → 9 → images → รายการแรก จะมี filename, subfolder และ type

แทนชื่อไฟล์ที่ได้จริงใน URL อย่าเอาชื่อจาก filename_prefix ถ้าระบบส่ง subfolder มาด้วย ให้ส่งค่าจริงแทนช่องว่างในตัวอย่าง

```
curl -fsS -G http://127.0.0.1:8188/view \
  --data-urlencode 'filename=YOUR_FILENAME.png' \
  --data-urlencode 'subfolder=' \
  --data-urlencode 'type=output' \
  -o generated-image.png
```

กราฟ API และ workflow JSON สำหรับแสดงบน canvas เป็นคนละรูปแบบ ไฟล์ comfy-sd15-api.json นี้ใช้เป็น HTTP body ไม่ใช่ไฟล์สำหรับเปิดบน canvas โดยตรง

อ้างอิง [16]

12 เปิดใช้ครั้งต่อไปและสำรองข้อมูล

เปิดใช้ตามลำดับ

เปิด Docker Desktop และ Ollama แล้วเริ่ม n8n จากโฟลเดอร์เดิม ถ้าจะสร้างภาพให้เปิด ComfyUI instance ด้วย หลังรีสตาร์ท Mac ต้องรอ Docker Engine พร้อมก่อน

```
cd ~/local-ai-stack
docker compose up -d
docker compose ps
```

หยุดเมื่อไม่ได้ใช้

```
docker compose stop
ollama stop qwen3:4b
```

หยุด ComfyUI ผ่าน instance manager หรือปิดแอป คำสั่ง `compose stop` หยุด n8n โดยคงข้อมูลเดิมไว้ใน Docker Desktop อาจเปิด Resource Saver ให้เหมาะกับการใช้งาน

สำรอง n8n แบบครบทั้งฐานข้อมูลและ encryption key

รันขณะ Docker เปิดอยู่ คำสั่งนี้หยุด n8n ก่อนสำรอง SQLite แล้วสร้างไฟล์ลง backups ในโฟลเดอร์โปรเจกต์ ตั้งชื่อไฟล์ใหม่ถ้าจะเก็บหลายวัน

```
cd ~/local-ai-stack
mkdir -p backups
docker compose stop n8n
docker run --rm \
  -v mac-local-ai-n8n-data:/data:ro \
  -v "$PWD/backups:/backup" \
  alpine:3.22 \
  tar -czf /backup/n8n-backup.tar.gz -C /data .
docker compose start n8n
```

ตรวจว่าไฟล์ `backups/n8n-backup.tar.gz` มีอยู่ สำรอง `compose.yaml`, ไฟล์ workflows และโฟลเดอร์ภาพไว้ด้วย เก็บ backup ในที่ที่ควบคุมสิทธิ์ เพราะอาจมีข้อมูลและ credentials

ก่อนอัปเดต

สำรองข้อมูลก่อน เปลี่ยนเลขรุ่นใน `compose.yaml` เป็น stable ที่ตรวจ release notes แล้วใช้ `docker compose pull` และ `docker compose up -d` สำหรับ ComfyUI ให้สร้าง snapshot ก่อนเปลี่ยนรุ่นหรือเพิ่ม custom nodes

อย่าใช้ `docker compose down -v` หรือ Docker Factory Reset หากต้องการเก็บข้อมูล การย่อน image รุ่นเก่าอย่างเดียวยังไม่เพียงพอเมื่อฐานข้อมูลเปลี่ยน schema

อ้างอิง [7], [9], [18]

13 แก้ปัญหาที่พบบ่อย

อาการ	วิธีตรวจและแก้
ollama command not found	เปิด Ollama อีกครั้งและทำขั้นตอนติดตั้ง CLI เปิด Terminal ใหม่ หรือใช้พารามิเตอร์ตามคำสั่งด้านล่าง
Address already in use 11434	แอป Ollama อาจเปิด server อยู่แล้ว ใช้แอปเดิมและไม่เปิด ollama serve ซ้ำ
n8n เปิด localhost ไม่ได้	ตรวจ docker compose ps และ logs รวมถึงเปิด Host networking แล้วหรือยัง
n8n เรียก AI ไม่ได้ แต่ Mac เรียกได้	ทดสอบจาก container ตามบท 6 และตรวจ Host networking ใช้ 127.0.0.1 แทน localhost ใน HTTP Request
model not found	ตรวจ ollama list แล้วแก้ไขชื่อโมเดลให้ตรงทุกตัวอักษร
ComfyUI ไม่พบ checkpoint	เปิดโฟลเดอร์ Shared Models จริง วางลง checkpoints แล้ว Refresh หรือ restart
ComfyUI ภาพไม่ออกหรือ RAM เต็ม	ใช้ 512 x 512, batch 1 ปิดแอปหนัก ปล่อย Ollama ด้วย ollama stop แล้วลองกราฟมาตรฐาน
MPS error หรือ custom node ใช้ไม่ได้	กลับไปทดสอบ SD 1.5 กับ node มาตรฐาน ตรวจ log และอัปเดตผ่าน instance manager; node ที่ต้องใช้ CUDA ไม่ได้แปลว่าใช้บน Mac ได้
workflow ภาพ timeout	ดูคิวและ log ของ ComfyUI เช็กพอร์ตและไฟล์โมเดล ก่อนเพิ่มเวลารอหรือส่งงานใหม่

```
/Applications/Ollama.app/Contents/Resources/ollama --version
curl -fsS http://127.0.0.1:11434/api/tags
curl -fsS http://localhost:5678/healthz
curl -fsS http://127.0.0.1:8188/system_stats
```

Mac RAM 16 GB ควรให้ AI ข้อความและภาพทำงานตามลำดับ เริ่มด้วยงานที่ละชิ้น ความเร็วจริงขึ้นกับชิป RAM รุ่นโมเดล ความละเอียด และโปรแกรมอื่นที่เปิดอยู่

localhost หมายถึงเครื่องที่โปรแกรมกำลังรัน ถ้าใช้ n8n Cloud หรือ Docker แบบ bridge จะไม่ใช่เครือข่ายเดียวกับตัวอย่างนี้ อย่าสลับวิธีติดตั้งระหว่างทาง

อ้างอิง [1], [4], [6], [13], [15], [16]

14 ตรวจสอบความพร้อมและใช้งานต่อ

สิ่งที่ตรวจ	ผลที่ต้องเห็น
Ollama CLI	ollama list แสดง qwen3:4b และตอบคำถามได้
Ollama API	/api/chat ตอบ JSON และมี message.content
Docker และ n8n	compose ps แสดงบริการทำงาน และเปิด localhost:5678 ได้
n8n ไป Ollama	คำสั่งใน container อ่าน /api/tags ได้
ComfyUI	SD 1.5 สร้างภาพ 512 x 512 ได้ และ /system_stats ตอบ
workflow ครบสามตัว	Download Image มี binary data และมีไฟล์ภาพใน output
ข้อมูลหลังหยุดแล้วเปิดใหม่	บัญชีและ workflow n8n ยังอยู่
backup	สร้างไฟล์สำรองได้และเก็บแยกจากข้อมูลทำงาน

เริ่มต่อยอดจากงานเล็ก

ลองให้ n8n สรุปข้อความ เขียนแคปชัน หรือเตรียมพรีมพ์สินค้า เปลี่ยน Manual Trigger เป็น Schedule Trigger เมื่อตรวจผลได้แล้ว และตั้งเวลาเป็น Asia/Bangkok

Mac และแอปที่เกี่ยวข้องต้องทำงานอยู่ หากเครื่องหลับหรือปิด เครื่องจะไม่ทำงานตามเวลา การตั้ง restart ช่วยเริ่ม container เมื่อ Docker กลับมาทำงาน แต่ไม่ได้เปิดเครื่องให้เอง

สำหรับ webhook จาก LINE หรือบริการภายนอก localhost รับคำขอจากอินเทอร์เน็ตไม่ได้ ต้องออกแบบ HTTPS endpoint หรือระบบ relay เพิ่มก่อน ซึ่งอยู่นอกขั้นติดตั้ง Local ในคู่มือนี้

ขอบเขตการตรวจคู่มือ

ตรวจขั้นตอนกับเอกสารทางการ ตรวจโครงสร้างไฟล์ Compose และ JSON ตัวอย่าง รวมถึงรูปแบบ API แล้ว คู่มือนี้ยังไม่ได้ทดสอบติดตั้งและสร้างภาพบน Mac mini เครื่องจริงของผู้อ่าน จึงมีจุดตรวจผลให้ทำตามแต่ละบท

ไฟล์ตัวอย่างเป็น workflow สำหรับเริ่มทดลอง ไม่ได้เผยแพร่หรือส่งข้อความให้ใคร เมื่อจะเชื่อมข้อมูลจริงให้ตรวจคำตอบและ node ปลายทางตามงานของคุณ

อ้างอิง [3], [6], [9], [16]

15 แหล่งอ้างอิงทางการ

ตรวจสอบวันที่ 3 ตุลาคม 2026 เอกสารออนไลน์และชื่อเมนูอาจเปลี่ยนได้ ใช้ลิงก์ด้านล่างเพื่อดูรายละเอียดของขั้นตอนที่คุณกำลังทำ

- [1] Ollama บน macOS
- [2] โมเดลตัวอย่าง Qwen3 4B
- [3] Ollama Chat API
- [4] Ollama FAQ และ GPU บน Mac
- [5] ติดตั้ง Docker Desktop บน Mac
- [6] Docker Host networking
- [7] n8n Docker และรุ่น stable ที่หน้าเอกสารระบุ
- [8] แนวทาง Docker Compose ของ n8n
- [9] ตัวแปรการตั้งค่า n8n
- [10] HTTP Request node
- [11] Ollama credentials ใน n8n
- [12] Import และ Export workflow
- [13] ติดตั้ง Comfy Desktop บน Mac
- [14] สร้าง ComfyUI instance
- [15] สร้างภาพแรกใน ComfyUI
- [16] ComfyUI API
- [17] โฟลเดอร์และการตั้งค่า Comfy Desktop
- [18] สำรองและกู้คืน n8n
- [19] ข้อกำหนดการติดตั้ง n8n ด้วย npm
- [20] โมเดล SD 1.5 FP16 จากลิงก์ของ Comfy Org

คู่มือนี้ปรับแนวทางจากเอกสารให้เป็นชุด Local ขนาดเล็ก: n8n ตัวเดียวบน Docker Compose ใช้ Host networking และ volume ถาวร ส่วนการรันโมเดลใช้แอปบน macOS ไม่ได้คัดลอกชุด Assistant หรือ sandbox ทั้งชุดจากเอกสาร n8n